NSImage in Snow Leopard

Session 111

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

NSImage in Snow Leopard

Ken Ferry
Cocoa Frameworks

NSImage in Snow Leopard

Lots of love

- Simplicity
- Performance
- Integration with other frameworks

Only on Mac OS

Agenda

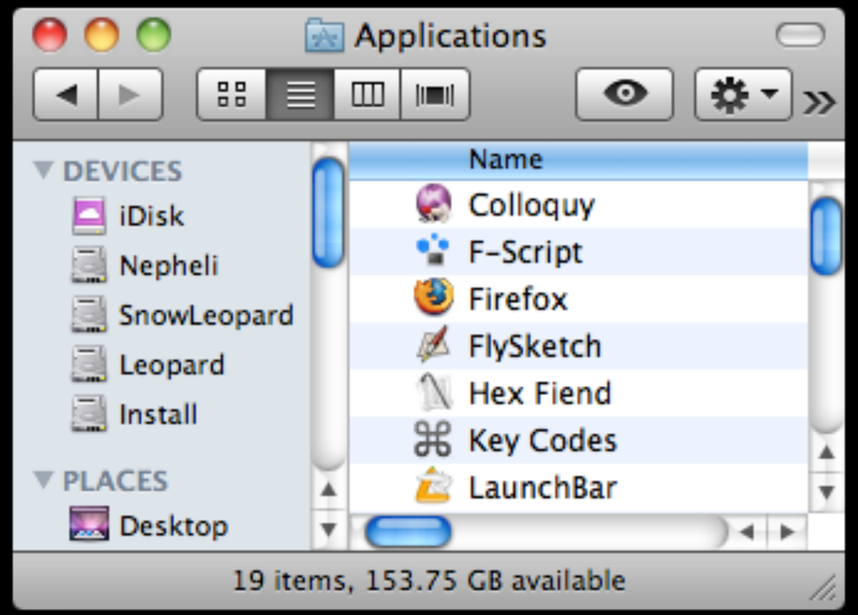
Core Architecture

Performance

The System as a Whole

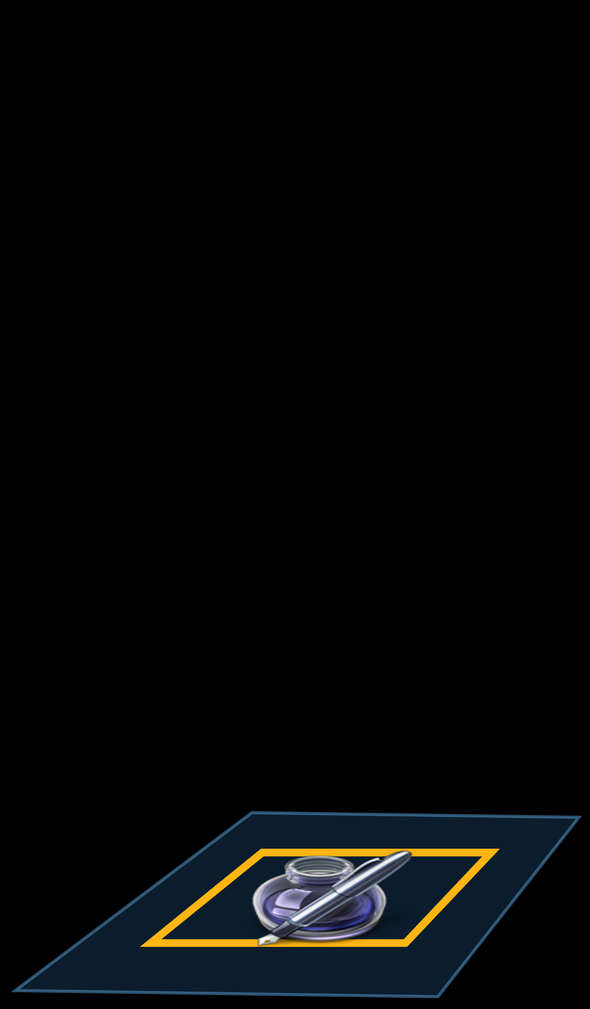
Images

User Data



Interface

Drawing



Representation Selection

Changes for 10.6

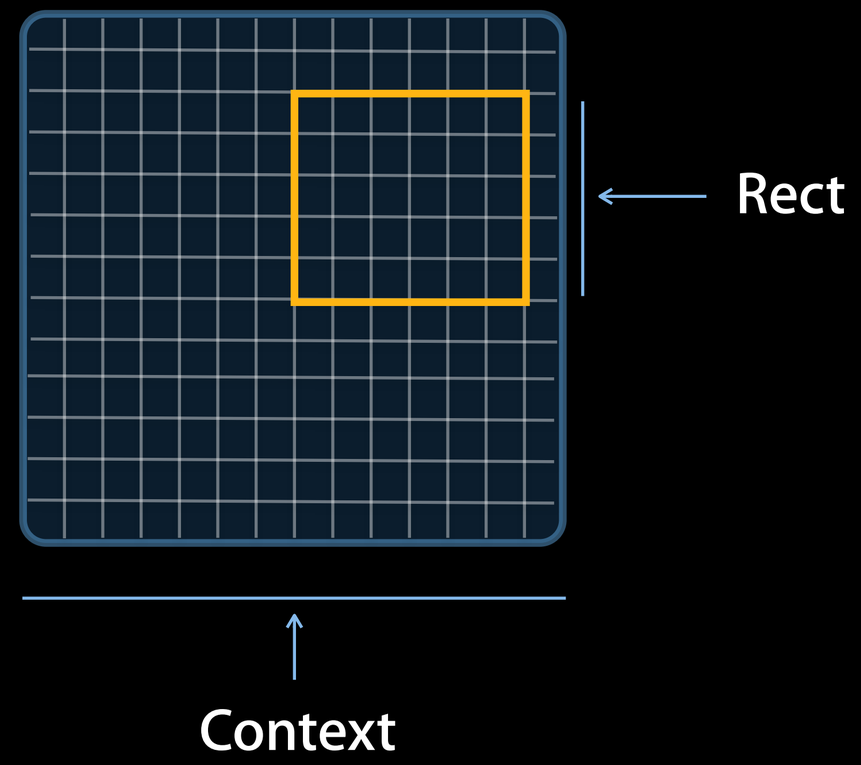
10.5 `-[NSImage bestRepresentationForDevice:]`

10.6 `-[NSImage bestRepresentationForRect:context:hints:]`

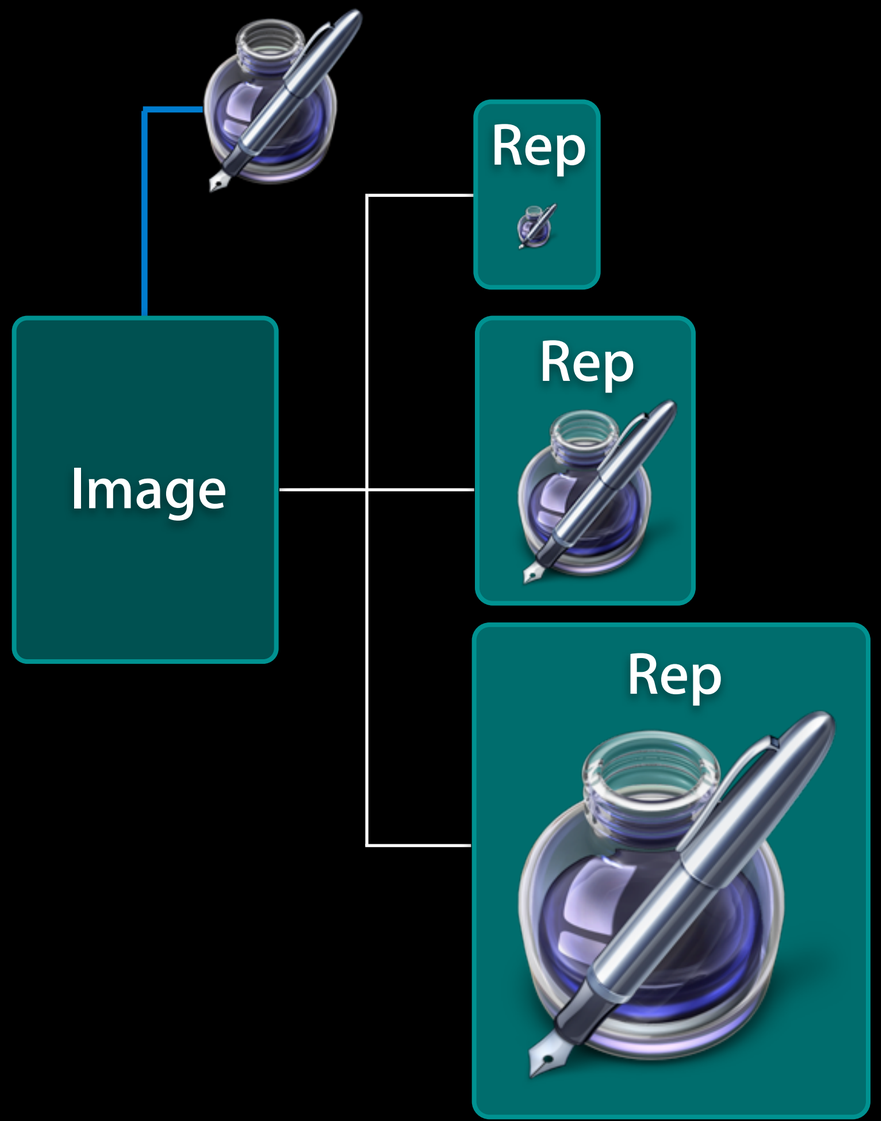
Representation Selection

The (rect, context, hints) tuple

- Pixels covered?
- Gray? Color?
- 16-bit? 8-bit? 1-bit?

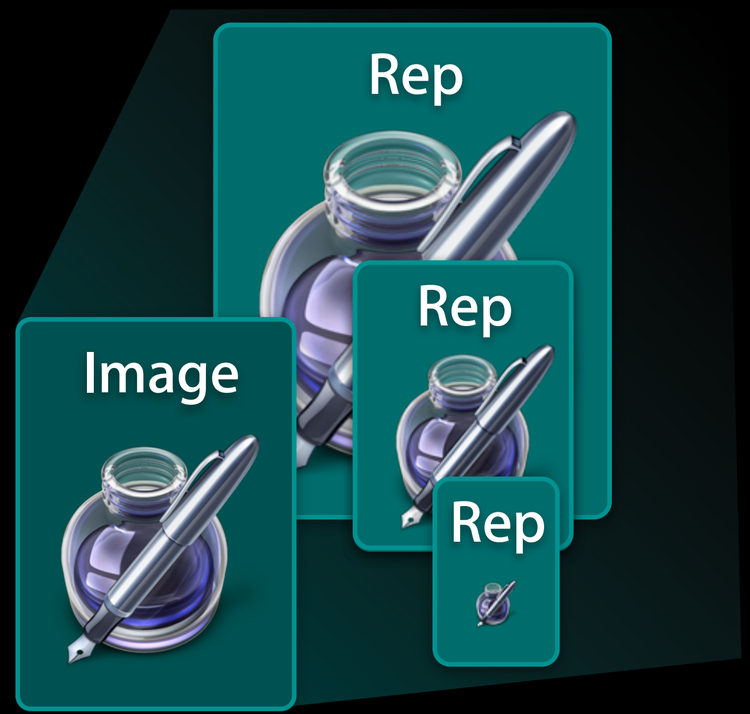


Caching



Caching

An optimized snapshot



Context

Caching

Changes for 10.6

Configuration affects behavior

Configuration affects only performance

Caching

No longer needed

```
- (void)setData
- (BOOL)isDataRe
- (void)setCached
- (BOOL)isCachedSep
- (void)setCacheDepth
- (BOOL)cacheDepthMat
- (void)setCacheMode
- (NSImageCacheMode
- (void)setScales
- (BOOL)scalesW
```

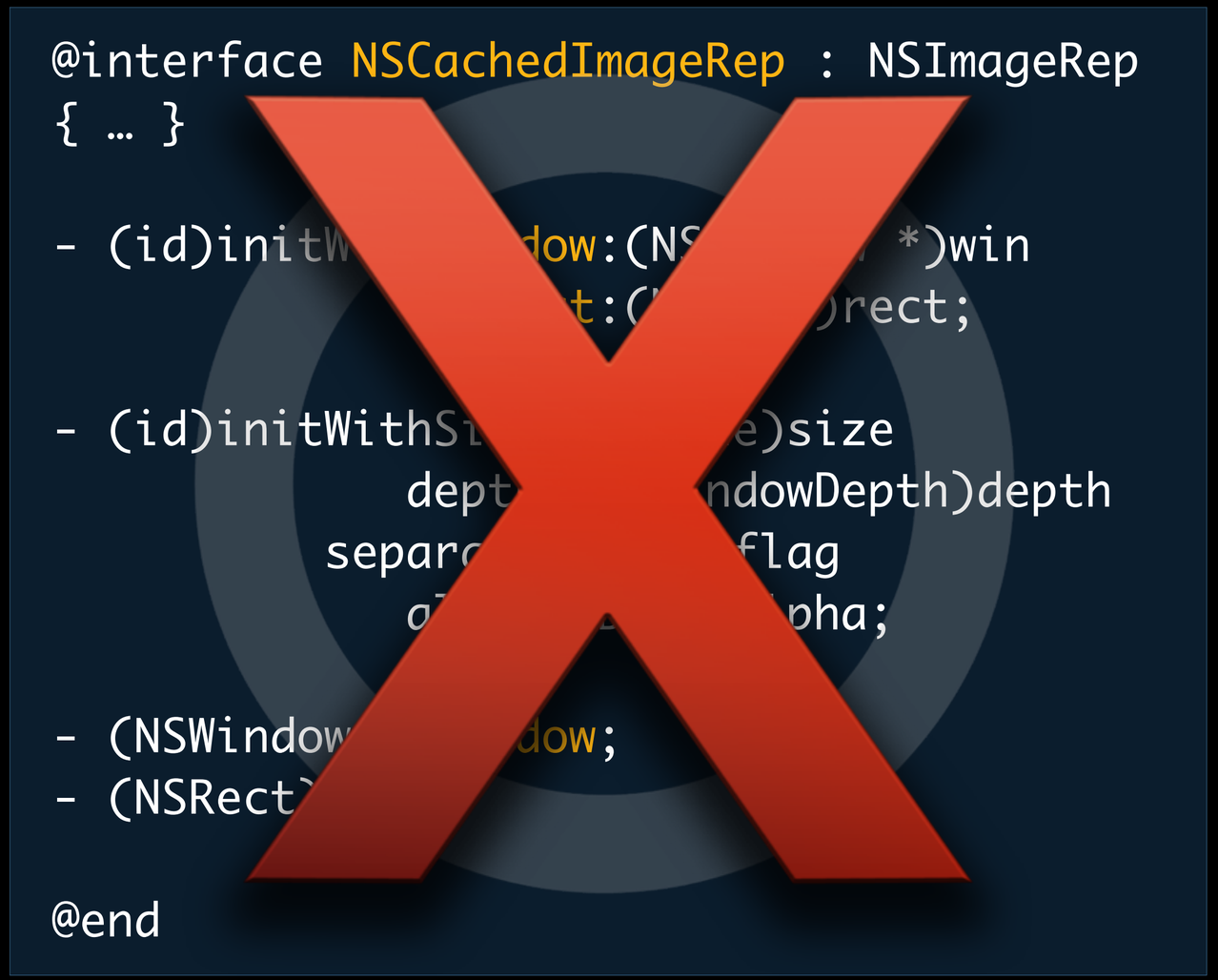
Caching

More changes for 10.6

Cache is an offscreen window

Cache is a CGImage

No More Windows



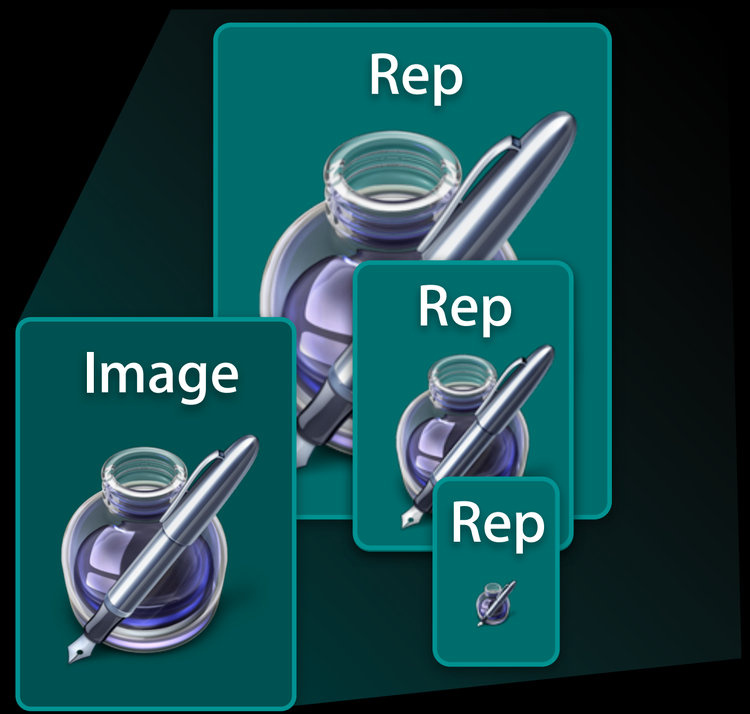
CGImage APIs

```
-[NSImage CGImageForProposedRect:context:hints:];
```

```
-[NSImage imageWithCGImage:size:];
```

CGImage and NSImage

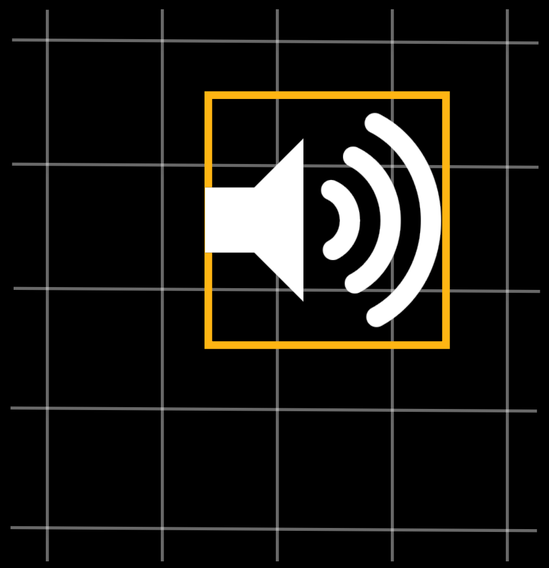
A snapshot



Context

CGImage and NSImage

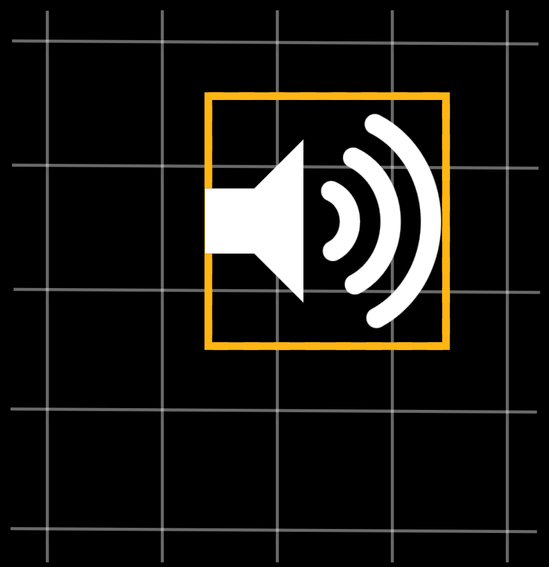
The proposedRect is an in-out parameter



```
NSRect destRect = …;
[image drawInRect:destRect
         fromRect:NSZeroRect
        operation:NSCompositeSourceOver
         fraction:1.0];
```

CGImage and NSImage

The proposedRect is an in-out parameter



```
context = [NSGraphicsContext currentContext];
NSRect destRect = …;
cgImage = [image CGImageForProposedRect:&destRect
                                context:context
                                  hints:nil];
CGContextDrawImage([context graphicsPort], destRect, cgImage);
```

The (rect, context, hints) Tuple

What if I don’t have anything to pass?

```
[image CGImageForProposedRect:     context:    hints:   ]
```

The (rect, context, hints) Tuple

What if I don’t have anything to pass?

```
[image CGImageForProposedRect:NULL context:nil hints:nil]
```

Smallest pixel integral rect containing {{0,0}, [image size]}

Default context for window on main display

No overrides of context parameters

NSImage and CGImage

Contract terms

Drawing the CGImage in the given rect, in the given context, is equivalent to drawing the NSImage in that rect in the context.

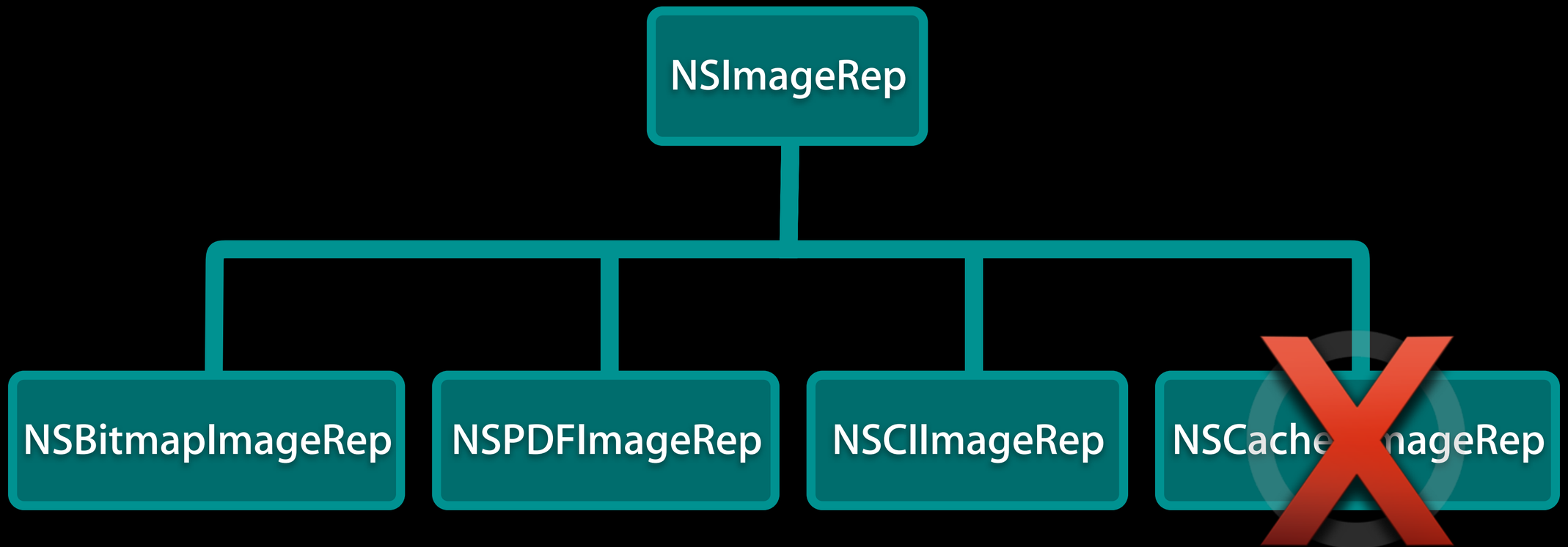
Does This Seem Kind of Expensive?

We’re not done yet

```
-[NSImageRep CGImageForProposedRect:context:hints:];
```

NSImageRep

What is it again?



NSImageRep

Built for subclassing

-[NSView drawRect:]

-[NSImageRep draw]

```
-[NSImageRep colorSpaceName]
-[NSImageRep bitsPerSample]
-[NSImageRep pixelsWide]
-[NSImageRep pixelsHigh]
```

NSImageRep

What about the other methods?

- Best effort only
- Optional

NSImageRep

Our new method

```
-[NSImageRep CGImageForProposedRect:context:hints:]
```

- Override to return a pre-existing CGImage
  - …or don’t
  - NSBitmapImageRep does!

NSBitmapImageRep

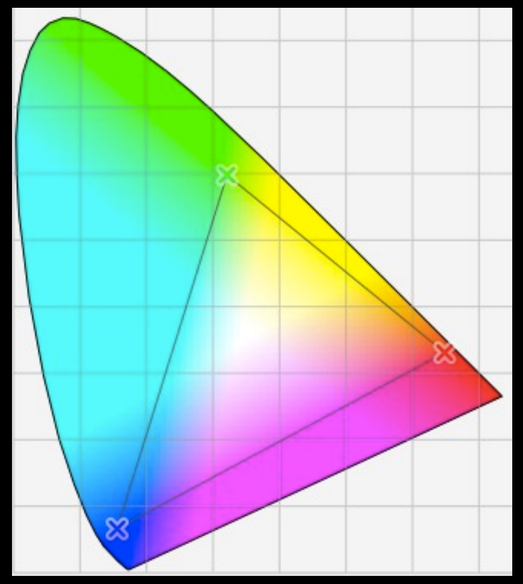
What is it?

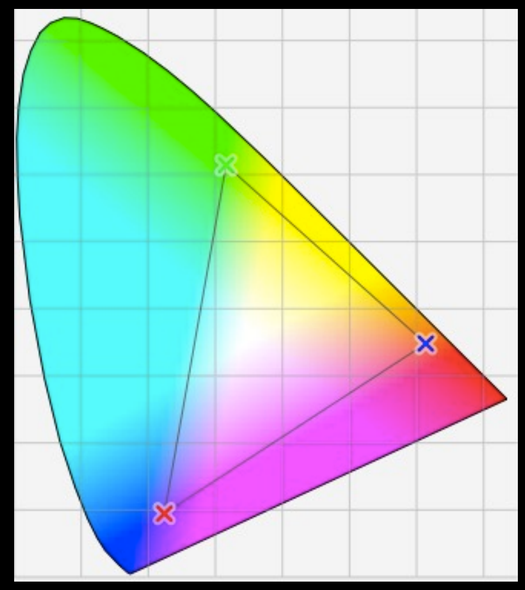
- Rectangular array of colors
- …like NSString is an array of characters
  - Similar difficulties in looking directly at the bytes

NSBitmapImageRep

Colorspaces

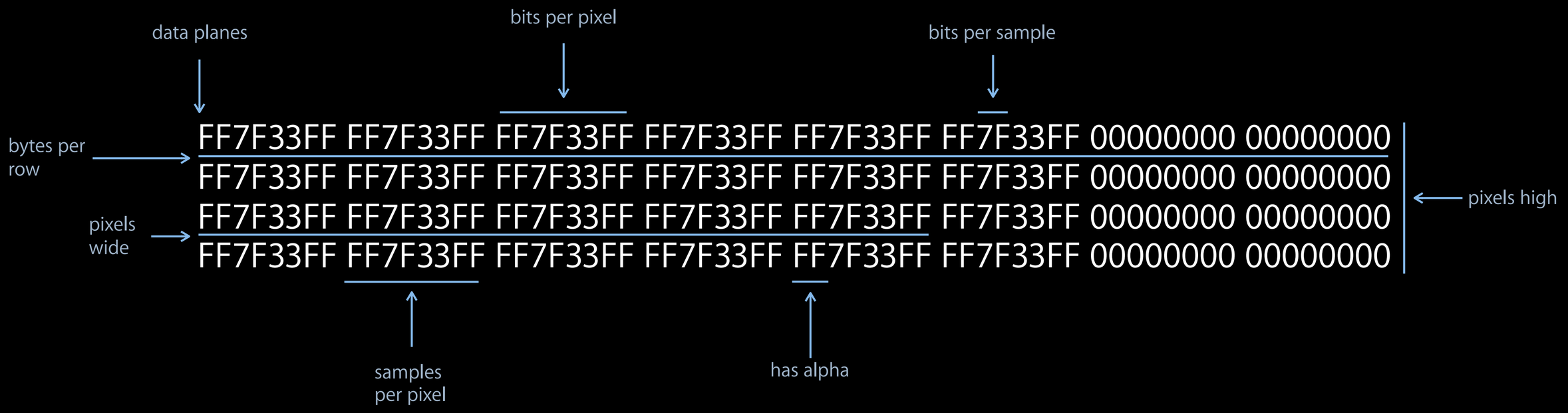
(0.5, 0.2, 1.0, 1.0)





NSBitmapImageRep

Pixel formats




```
-[NSBitmapImageRep initWithBitmapDataPlanes:pixelsWide:pixelsHigh:
    bitsPerSample:samplesPerPixel:hasAlpha:isPlanar:
    colorSpaceName:bitmapFormat:bytesPerRow:bitsPerPixel:]
```

NSBitmapImageRep and CGImage

Very similar objects

```
-[NSBitmapImageRep CGImage]
-[NSBitmapImageRep initWithCGImage:]
```

NSBitmapImageRep stores CGImage, not data

End-to-End Impedance Matching

Implications

- CGImages preserved from start to finish
  - Less packing, unpacking, processing of data
- Caching, snapshotting, often very cheap

Mutability

A remaining difference from Quartz

- Quartz separates source from destination; Cocoa doesn't
  - Image and bitmap both
- Internally, copy on write

Mutability

Recommended approach

- Uncontrolled mutation
  - Affects remote parts of the app
    - setSize
  - Thread safety
- Init, mutate, vend
  - Copy, mutate, vend

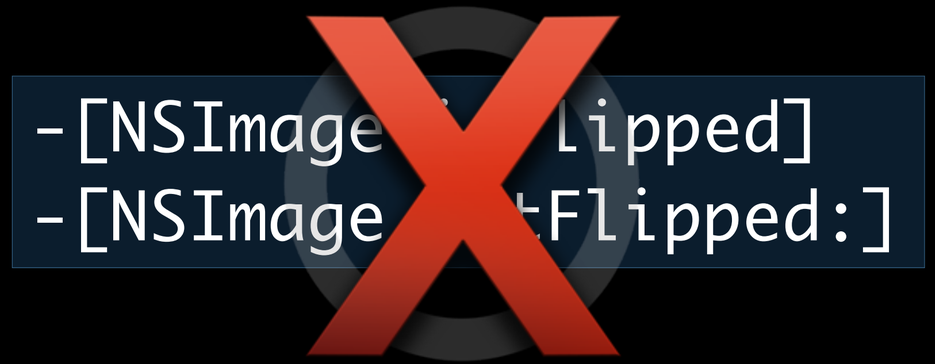
Methods to Watch Out For

Poor misunderstood methods

`-[NSImage setFlipped:]`	Deprecated
`-[NSImage lockFocus]`	Replaces reps with new backing
`-[NSBitmapImageRep bitmapData]`	Must know the pixel format
`-[NSImage TIFFRepresentation]`	For persistence, not making an NSBitmapImageRep

-[NSImage setFlipped:]

Drawing in a flipped context




```
-[NSImage drawInRect:
              fromRect:
             operation:
              fraction:
        respectFlipped:
                 hints:]
```

Older Drawing Methods

Methods to Watch Out For

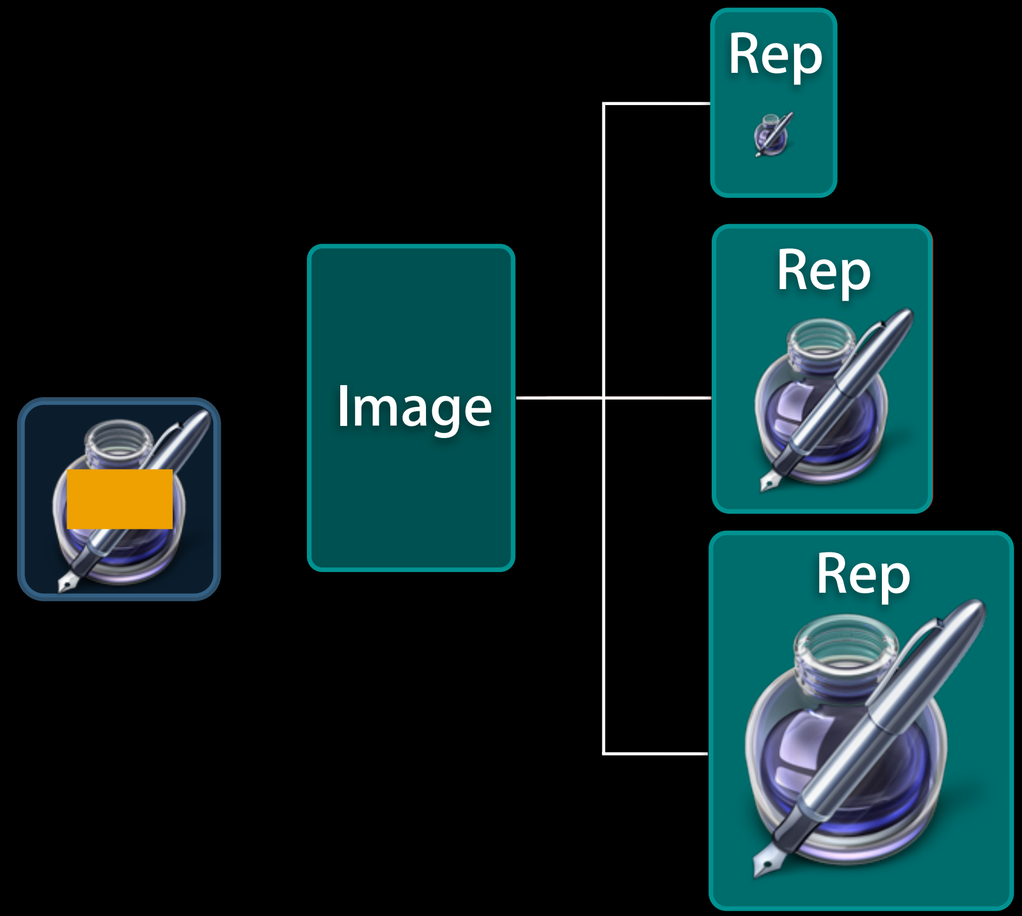
Poor misunderstood methods

`-[NSImage setFlipped:]`	Deprecated
`-[NSImage lockFocus]`	Replaces reps with new backing
`-[NSBitmapImageRep bitmapData]`	Must know the pixel format
`-[NSImage TIFFRepresentation]`	For persistence, not making an NSBitmapImageRep

-[NSImage lockFocus]

Drawing into an NSImage

```
[image lockFocus];
[NSRectFill(rect);
[image unlockFocus];
```



-[NSImage lockFocus]

Changes for 10.6

Drawing was inside of a view in a window.
[[NSView focusView] isFlipped] could be YES.

No window, no view involved.
Context is window-like.

-[NSImage lockFocus]

Conclusions

- Audit your code for uses of +[NSView focusedView] between lockFocus and unlockFocus
- -[NSImage lockFocus] is
  - Destructive and creates new memory
  - Fine to use when appropriate, but be aware
- Best for drawing, destined for the screen
  - Not great for persistent images

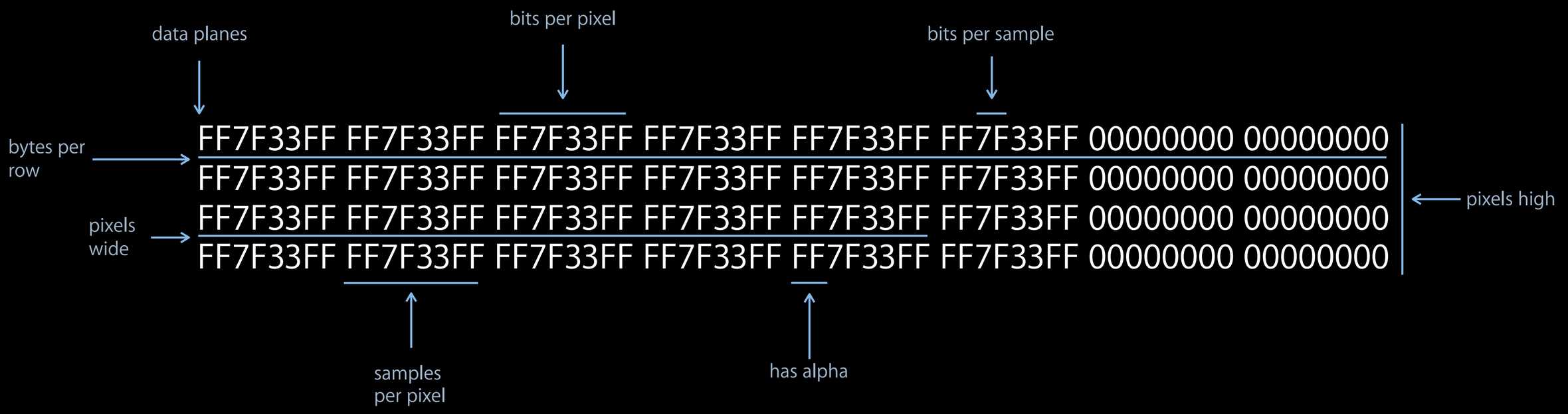
Methods to Watch Out For

Poor misunderstood methods

`-[NSImage setFlipped:]`	Deprecated
`-[NSImage lockFocus]`	Replaces reps with new backing
`-[NSBitmapImageRep bitmapData]`	Must know the pixel format
`-[NSImage TIFFRepresentation]`	For persistence, not making an NSBitmapImageRep

Bitmap Data

Pixel formats are a problem




```
-[NSBitmapImageRep initWithBitmapDataPlanes:pixelsWide:pixelsHigh:
bitsPerSample:samplesPerPixel:hasAlpha:isPlanar:
colorSpaceName:bitmapFormat:bytesPerRow:bitsPerPixel:]
```

Bitmap Data

Specify the pixel format

```
bitmap = [[NSBitmapImageRep alloc] initWithBitmapDataPlanes:NULL pixelsWide:width
            pixelsHigh:height bitsPerSample:8 samplesPerPixel:32 hasAlpha:YES isPlanar:NO
            colorSpaceName:NSCalibratedRGBColorSpace bitmapFormat:NSAlphaFirstBitmapFormat
            bytesPerRow:0 bitsPerPixel:32];

[NSGraphicsContext saveGraphicsState];
[NSGraphicsContext setContext:[NSGraphicsContext graphicsContextWithBitmapImageRep:bitmap]];
<drawing goes here>
[NSGraphicsContext restoreGraphicsState];
[bitmap bitmapData];
```

Methods to Watch Out For

Poor misunderstood methods

`-[NSImage setFlipped:]`	Deprecated
`-[NSImage lockFocus]`	Replaces reps with new backing
`-[NSBitmapImageRep bitmapData]`	Must know the pixel format
`-[NSImage TIFFRepresentation]`	For persistence, not making an NSBitmapImageRep

TIFFRepresentation

For persisting data

```
[[image TIFFRepresentation] writeToFile:path atomically:YES];
```

```
[[NSBitmapImageRep alloc] initWithData:[image TIFFRepresentation]];
```

TIFFRepresentation

So how ought I make a bitmap?

- Draw in a bitmap
- CGImage APIs

Final Thoughts on Bitmap Data

Do you really need it?

- Drawing is fast
- Consider CoreImage
- Draw to a format you understand

Agenda

Core Architecture

Performance

The System as a Whole

Performance

Same drawing, same image

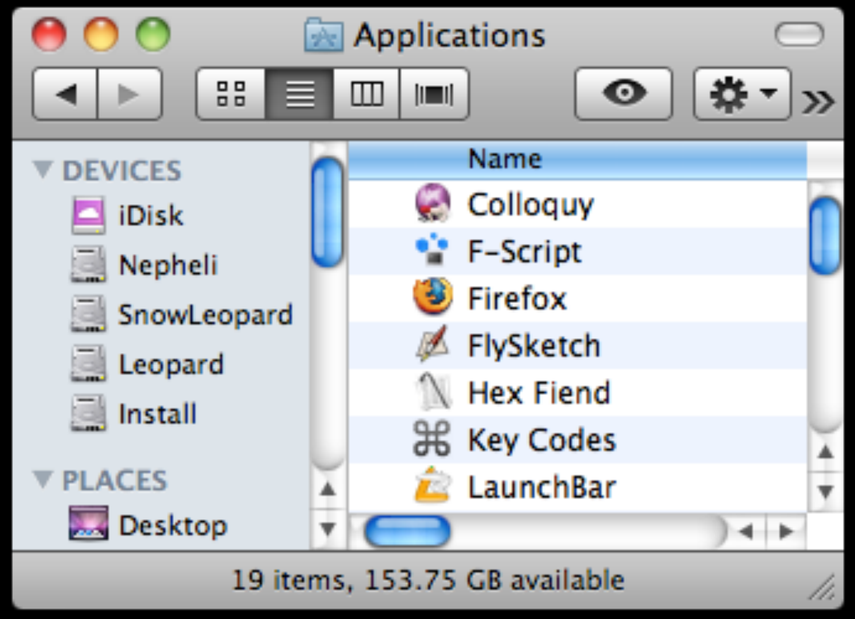


Image Performance

Tips, tricks, and debugging techniques

What Did We Learn?

Basic lessons

- Same drawing, same image
  - Different drawing, different images
- Profile
  - NSNewBitmapBackingStore
- Use the cache
- Disable the cache
  - When caching is done above or below
  - For transient images

What Did We Learn?

Tips and tricks

- The name table doesn’t keep images alive
- Hit testing API
- Three- and nine-part drawing APIs
- initByReferencingURL does lazy IO
- Don’t be afraid of NSImageRep subclassing

Agenda

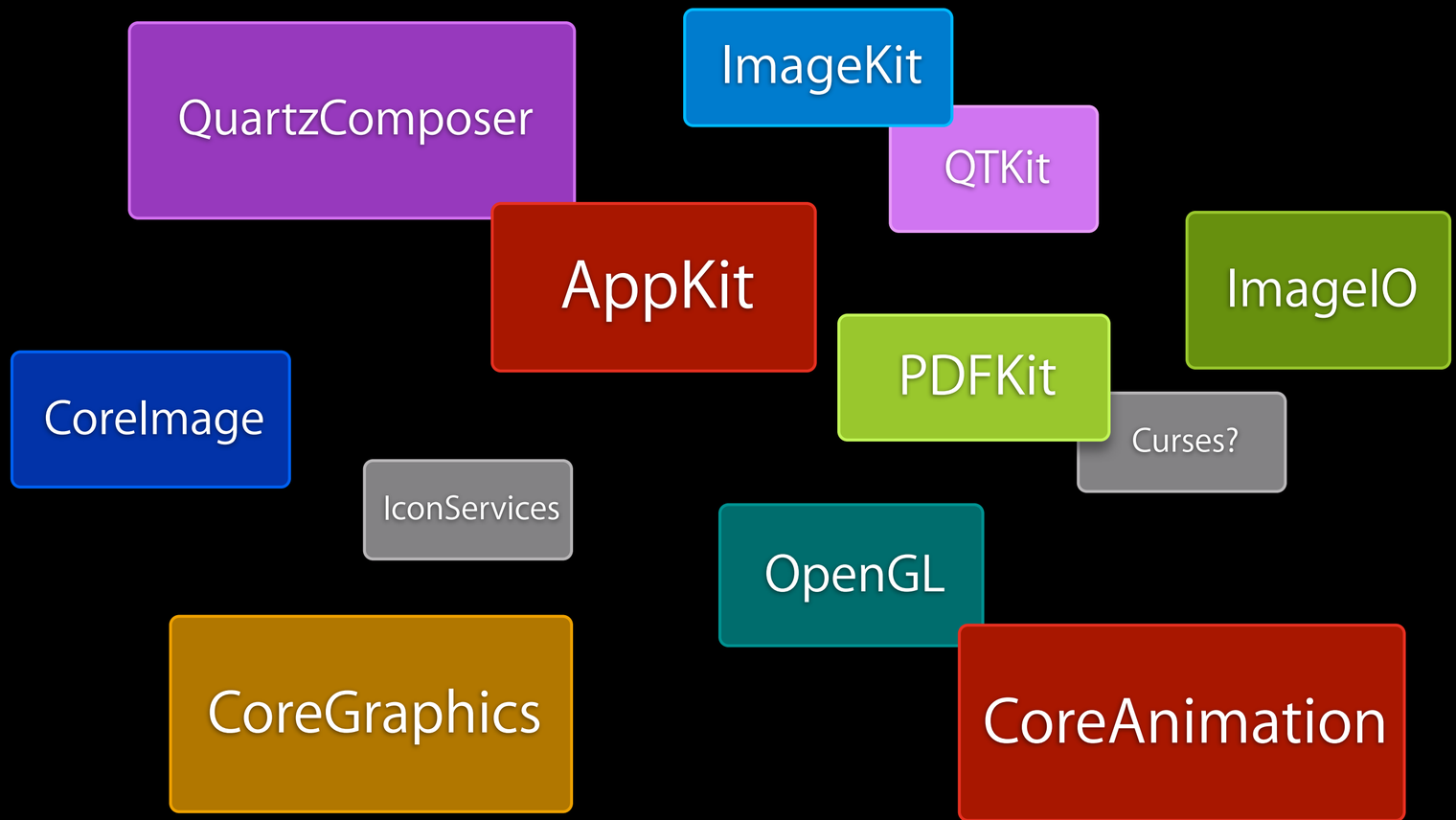
Core Architecture

Performance

The System as a Whole

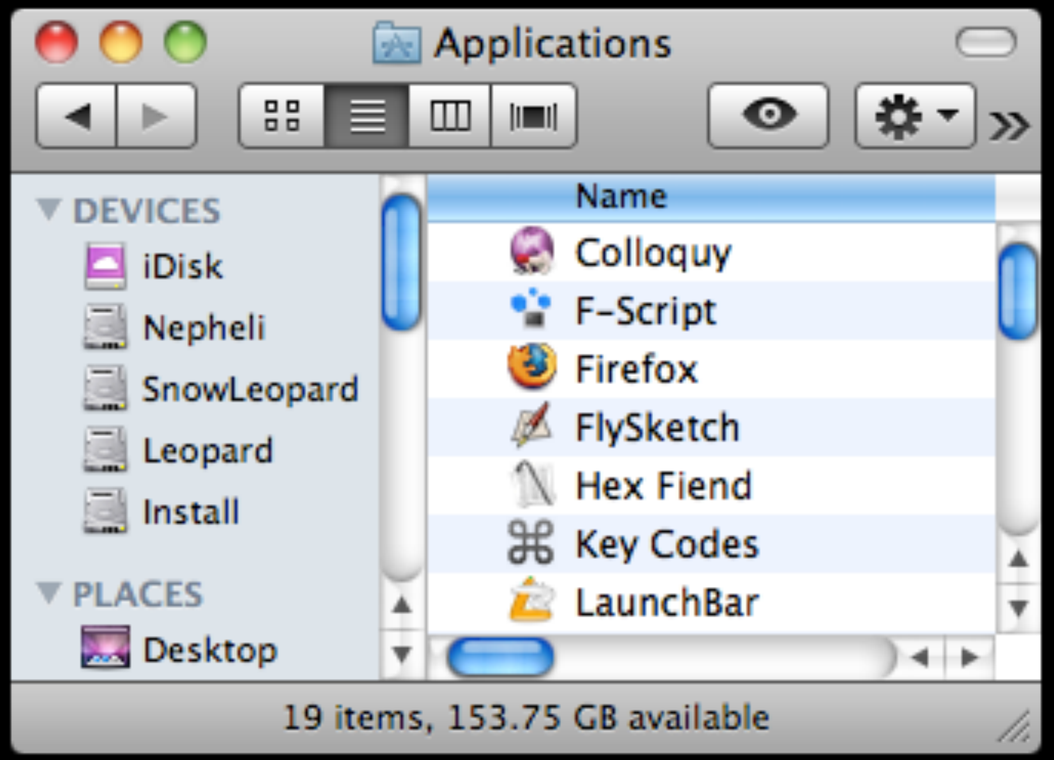
Graphics on Mac OS X

How does it all fit together?



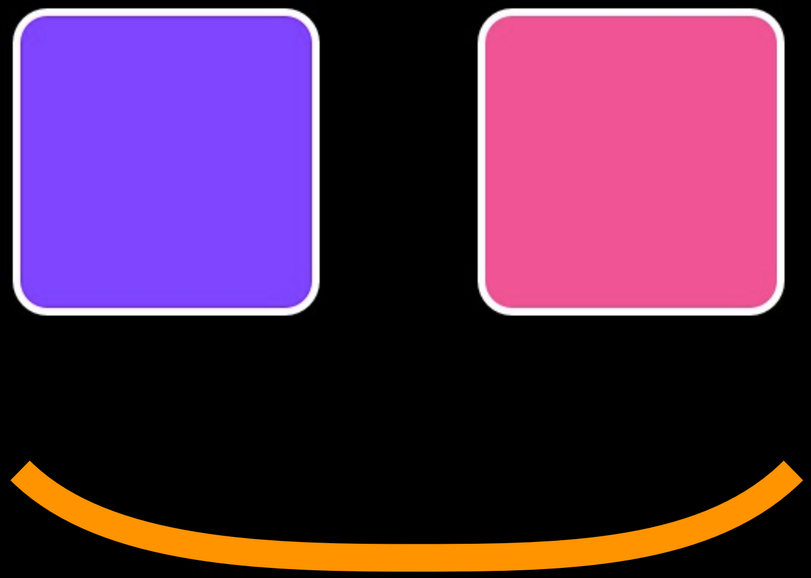
AppKit

Integration, default choice



Quartz

Foundation for drawing



CoreImage

Pixel processing

ImageIO

Bitmap file formats

ImageKit

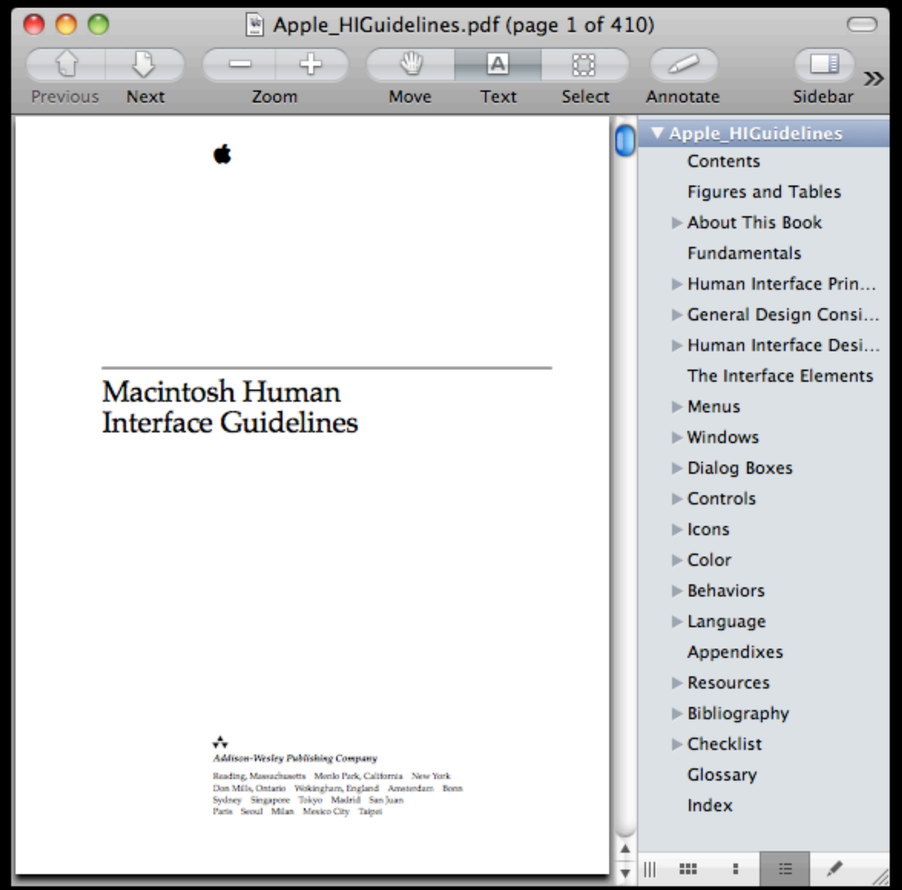
Standard UI panels

CoreAnimation

Declare a scene

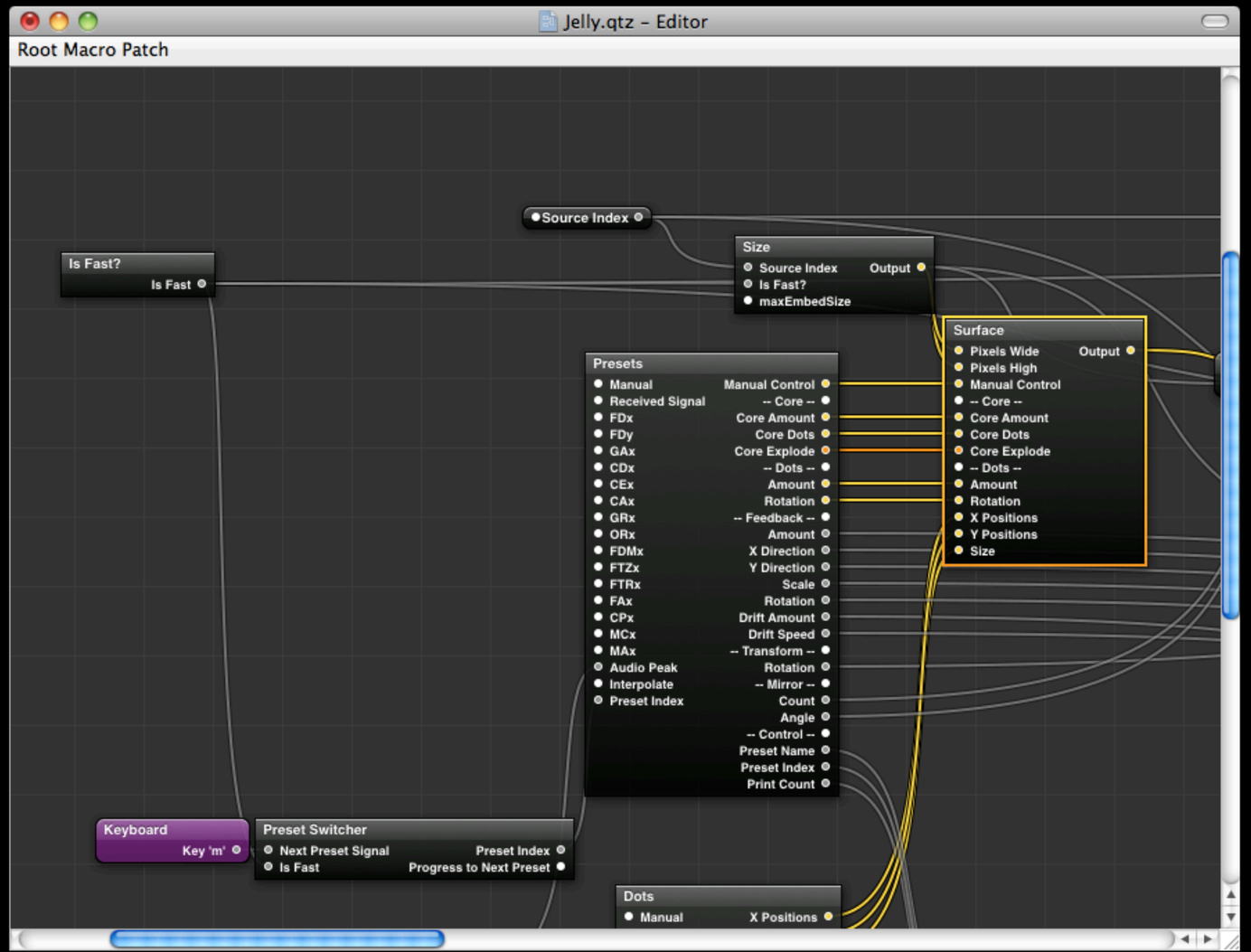
PDFKit

PDFs as documents, not images



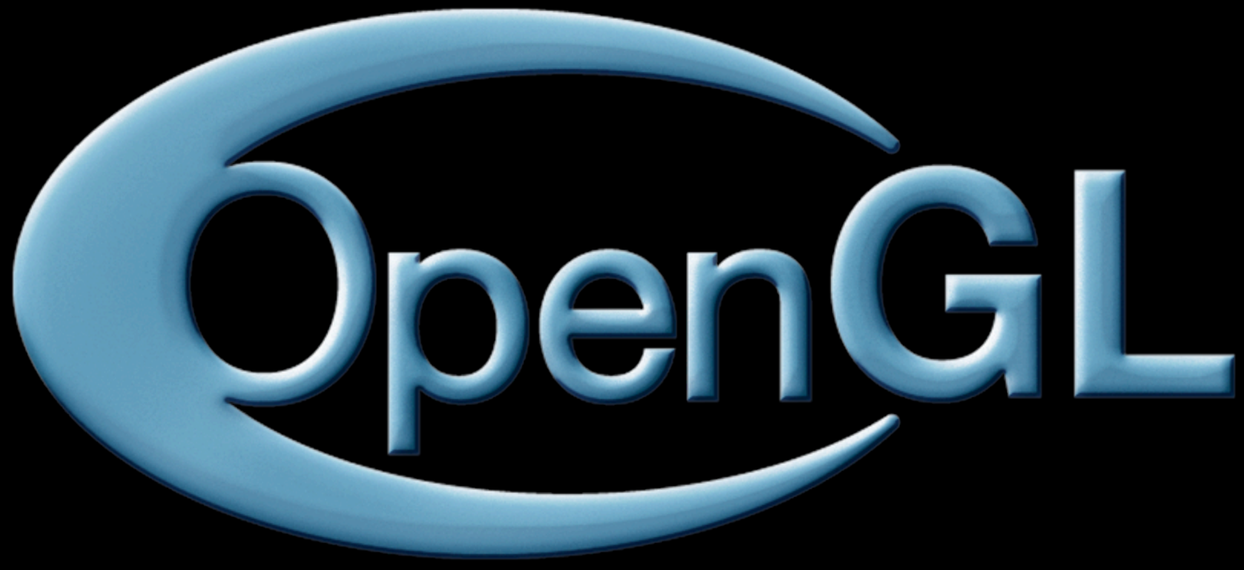
QuartzComposer

Visual programming environment



OpenGL

Cross-platform, 3D API



QTKit

Video, media

IconServices

Subsumed by NSImage, NSWorkspace, NSURL

Cheat Sheet

Framework	Function	When to Investigate
AppKit	Integration, general drawing	Unless you are in a more specific case; abstraction avoids conversions that hurt fidelity and performance
Core Graphics	Foundation layer for drawing	If a drawing-related task seems to be hard in AppKit; it's used by AppKit, but direct use is fine
CoreImage	Per-pixel image processing, a la Photoshop filters	For processing of image data
ImageIO	Reading and writing bitmap file formats, including metadata	If a bitmap file handling task seems to be hard in AppKit; it's used by AppKit, but direct use is fine
ImageKit	Standard user-facing panels and views	To let your user work with images; it's like NSColorPanel for images
Core Animation	Compositing a scene; similar to and integrates with the NSView hierarchy	If you need to optimize view hierarchy performance, as is often true for animation
PDFKit	Working with PDFs as documents	For any task that deals with PDFs as documents as opposed to as images
Quartz Composer	Complete visual programming environment	If visual programming sounds like fun; peer of AppKit, with integration; also nice for prototyping image processing
OpenGL	Cross-platform 2D and 3D accelerated graphics library	For cross-platform code and 3D scenes
QTKit	Multimedia	For movies
Icon Services	Working with file icons, and drawing system standard art	Legacy; prefer NSImage, NSURL, NSWorkspace

We Made It!

What did we learn?

- NSImage
  - Working hard for the common man
    - Now with behavioral transparency!
    - Now with impedance matching!
- Performance
  - Same drawing, same image
- Graphics frameworks
  - Use the cheat sheet

More Information

Matt Drance
Cocoa Evangelist
mdrance@apple.com

AppKit Release Notes
Required reading
https://developer.apple.com/snowleopard/library/releasenotes/Cocoa/AppKit.html

Cocoa Drawing Guide
http://developer.apple.com/documentation/Cocoa/Conceptual/CocoaDrawingGuide/Introduction/Introduction.html

Q&A

WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09